sCO2_Loop_Calcs

October 3, 2021

IMPORTANT Ensure you are utilizing 64-bit REFPROP with 64-bit python. If using the free version of REFPROP (MINI-REFPROP), please use 32-bit python and make changes to match the location where MINI-REFPROP is installed and make changes to the REFPROPFunctionLibrary function to read the REFPROP.DLL file.

Information on REFPROP and functions can be found here: https://buildmedia.readthedocs.org/media/pdf/refprop-docs/latest/refprop-docs.pdf

0.0.1 IMPORT PACKAGES & FUNCTIONS

[1]:
```
# Dictate the environment's loctaion of REFPROP
import os
os.environ['RPPREFIX'] = r'C:/Program Files (x86)/REFPROP'
```

[2]:
```
# Import the main class from the Python library
from ctREFPROP.ctREFPROP import REFPROPFunctionLibrary

# Imports from conda-installable packages
import pandas as pd

# Import numpy
import numpy as np

# Import matplotlib for plotting
import matplotlib.pyplot as plt

# Import Math for common values such as PI
import math
```

[3]:
```
# Instantiate the library, and use the environment variable to explicitly state␣
↪which path we want to use.
# As mentioned above, this will be changed to call the correct REFPROP␣
↪functions to be used
# with MINI-REFPROP and 32-bit python.
# If using MINI-REFPROP and 32-bit python please make the following changes
# RP = REFPROPFunctionLibrary('C:/Program FIles (x86)/MINI-REFPROP\\REFPROP.
↪DLL')
RP = REFPROPFunctionLibrary(os.environ['RPPREFIX'])
```

[4]:
```python
# This will call which root directory that will be used for the program.
RP.SETPATHdll(os.environ['RPPREFIX'])
```

[5]:
```python
# Get the unit system we want to use (Mass base SI gives units in
# K, Pa, kg, m, N, J, W, and s)
MASS_BASE_SI = RP.GETENUMdll(0, "MASS BASE SI").iEnum
```

0.0.2 sCO2 Loop Calculations

System Parameters

[6]:
```python
# Tube inner diameter and outer diameter
Tube_OD = 0.50 # [inch]
Tube_Thick = 0.065 # [inch]
Tube_ID = Tube_OD - 2 * Tube_Thick # inch

Tube_OD = Tube_OD * 0.0254 # [Convert inches to meters]
Tube_ID = Tube_ID * 0.0254 # [Convert inches to meters]

# Mass flow rate of sCO2
m_dot = 0.2 # [kg/s]
```

Outlet of Compressor (State 1)

[7]:
```python
# Temperature will be compared at end of script
T1 = 60 # [C]
P1 = 2600 # [psia]

T1 = T1 + 273.15 # Convert C to Kelvin
P1 = P1 * 6894.8 # convert psia to Pa

print("Pressure at Outlet of Compressor =", P1/6894.8, "psia")
print("Temperature at Outlet of Compressor =" , (T1 - 273.15) * (9/5) + 32, "F")
```

```
Pressure at Outlet of Compressor = 2600.0 psia
Temperature at Outlet of Compressor = 140.0 F
```

[8]:
```python
# Obtain fluid properties from the pressure and temperature outlined above
State_1 = RP.REFPROPdll("CO2","PT","D;V;H;S;CP/CV;W;TCX;VIS;PRANDTL",␣
↪MASS_BASE_SI,0,0,P1,T1,[1.0])

# Outputs will be placed into data frame for organization
State_1 = pd.DataFrame(State_1.Output[0:9],
            index = ['Density [kg/m^3]', 'Volume [m^3/kg]', 'Enthalpy [J/kg]',␣
↪'Entropy [J/kg]',
                     'CP/CV', 'Speed of Sound', 'Thermal Cond. [W/(mK)]',␣
↪'Viscosity [Pa-s]', 'Prandtl'],
            columns = ['State 1'])
```

```
# Display the data frame
State_1
```

[8]:

	State 1
Density [kg/m^3]	685.702611
Volume [m^3/kg]	0.001458
Enthalpy [J/kg]	331011.948444
Entropy [J/kg]	1374.687634
CP/CV	2.953841
Speed of Sound	373.917781
Thermal Cond. [W/(mK)]	0.074005
Viscosity [Pa-s]	0.000055
Prandtl	2.058781

Pressure drop towards Heat Source

[9]:
```
# Find Velocity, Reynolds Number, and darcy friction factor (assuming smooth␣
↪pipe)
Velocity = 4 * m_dot * State_1.loc['Volume [m^3/kg]','State 1'] / (math.pi *␣
↪Tube_ID**2)
Reynolds = State_1.loc['Density [kg/m^3]','State 1'] * Velocity * Tube_ID /␣
↪State_1.loc['Viscosity [Pa-s]','State 1']
Darcy_f = (0.79 * math.log(Reynolds) - 1.64)**(-2)
```

[10]:
```
# Using Estimated Length of Tubing connecting compressor and Heat Source,
# find the amount of pressure drop caused by fanno flow
Length = 3.71475 # [meters]

# Force acted on the wall of tube
Force = math.pi * Tube_ID * Darcy_f * State_1.loc['Density [kg/m^3]','State 1']␣
↪* (Velocity**2) * Length / 8

# Dimensionless Friction factor
f_dim = 4 * Force / (P1 * math.pi * Tube_ID**2)

# Inlet Mach Number of length of tubing
Mach_inlet = Velocity / State_1.loc['Speed of Sound','State 1']

# Formulation used to calculate Pressure drop (found in Adv. Fluid Mechanics␣
↪Textbook)
A_eq = ((Mach_inlet**2) * (1 + ((State_1.loc['CP/CV','State 1'] - 1) / 2) *␣
↪(Mach_inlet**2))) \
        / ((1 + State_1.loc['CP/CV','State 1'] * (Mach_inlet**2) - f_dim)**2)

# Find the positive outcome to the biquadratic Mach number
Mach_outlet_1 = math.sqrt((-1 * (1 - 2 * A_eq * State_1.loc['CP/CV','State 1'])\
```

```
                             + ((1 - 2 * A_eq * (State_1.loc['CP/CV','State 1'] +␣
↪1))**0.5))\
                            / ((State_1.loc['CP/CV','State 1'] - 1) - 2 * A_eq *␣
↪State_1.loc['CP/CV','State 1']**2))

# Find Outlet pressure caused by fanno flow (frictional loss)
P2 = P1 * (1 + State_1.loc['CP/CV','State 1'] * (Mach_inlet**2) - f_dim) / (1 +␣
↪State_1.loc['CP/CV','State 1']\
                                                                                *␣
↪(Mach_outlet_1**2))

print("Pressure at Inlet of Heat Source =", round(P2/6894.8 , 3), "psia")
print("Temperature at Inlet of Heat Source =" , (T1 - 273.15) * (9/5) + 32, "F")
```

```
Pressure at Inlet of Heat Source = 2595.418 psia
Temperature at Inlet of Heat Source = 140.0 F
```

[11]:
```
# Find the enthalpy at the inlet of the Heat Source
Enth_2 = (State_1.loc['Enthalpy [J/kg]','State 1'] * (1 + ((State_1.loc['CP/
↪CV','State 1'] - 1) / 2) * Mach_inlet**2)) / \
            (1 + ((State_1.loc['CP/CV','State 1'] - 1) / 2) *␣
↪(Mach_outlet_1**2))

Enth_2 # [J/kg]
```

[11]: 331011.80397706

Inlet of Heat Source (State 2)

[12]:
```
#Pressure and temperature of fluid at inlet of Heat Source
P2 = P2



# Tube ID (using 1" OD Tubes)
Tube_HS_ID = .81 * 0.0254 # convert inches to meters

# Obtain fluid properties from the pressure and temperature outlined above
State_2 = RP.REFPROPdll("CO2","PH","T;D;V;S;CP/CV;W;TCX;VIS;PRANDTL",␣
↪MASS_BASE_SI,0,0,P2,Enth_2,[1.0])

# Outputs will be placed into data frame for organization
State_2 = pd.DataFrame(State_2.Output[0:9],
            index = ['Temperature [K]', 'Density [kg/m^3]', 'Volume [m^3/kg]',␣
↪'Entropy [J/kg]',
                     'CP/CV', 'Speed of Sound', 'Thermal Cond. [W/(mK)]',␣
↪'Viscosity [Pa-s]', 'Prandtl'],
            columns = ['State 2'])
```

```
# Display the data frame
State_2
```

[12]:

	State 2
Temperature [K]	333.106709
Density [kg/m^3]	685.353389
Volume [m^3/kg]	0.001459
Entropy [J/kg]	1374.825537
CP/CV	2.957623
Speed of Sound	373.487733
Thermal Cond. [W/(mK)]	0.073968
Viscosity [Pa-s]	0.000055
Prandtl	2.061120

[13]:

```
# Find Velocity, Reynolds Number, and darcy friction factor (assuming smooth␣
↪pipe)
Velocity = 4 * m_dot * State_2.loc['Volume [m^3/kg]','State 2'] / (math.pi *␣
↪Tube_HS_ID**2)
Reynolds = State_2.loc['Density [kg/m^3]','State 2'] * Velocity * Tube_HS_ID /␣
↪State_2.loc['Viscosity [Pa-s]','State 2']
Darcy_f = (0.79 * math.log(Reynolds) - 1.64)**(-2)
```

[14]:

```
# Using Estimated Length of Tubing used by Heat Source
# find the amount of pressure drop caused by fanno flow and Heat addition
Length = 19.2024 # [meters]

# Force acted on the wall of tube
Force = math.pi * Tube_HS_ID * Darcy_f * State_2.loc['Density [kg/m^3]','State␣
↪2'] * (Velocity**2) * Length / 8

# Dimensionless Friction factor
f_dim = 4 * Force / (P2 * math.pi * Tube_HS_ID**2)

# Heat Added into system (kW)
Q = 16

# Dimensionless heating factor
q_dim = Q * 1000 / (m_dot * Enth_2)

# Inlet Mach Number of length of tubing
Mach_inlet = Velocity / State_2.loc['Speed of Sound','State 2']

# Formulation used to calculate Pressure drop (found in Adv. Fluid Mechanics␣
↪Textbook)
A_eq = ((Mach_inlet**2) * (1 + ((State_2.loc['CP/CV','State 2'] - 1) / 2) *␣
↪(Mach_inlet**2) + q_dim)) \
        / ((1 + State_2.loc['CP/CV','State 2'] * (Mach_inlet**2) - f_dim)**2)
```

```
# Find the positive outcome to the biquadratic Mach number
Mach_outlet_1 = math.sqrt((-1 * (1 - 2 * A_eq * State_2.loc['CP/CV','State 2'])\
                          + ((1 - 2 * A_eq * (State_2.loc['CP/CV','State 2'] +␣
↪1))**0.5))\
                         / ((State_2.loc['CP/CV','State 2'] - 1) - 2 * A_eq *␣
↪State_2.loc['CP/CV','State 2']**2))

# Find Outlet pressure caused by fanno flow (frictional loss)
P3 = P2 * (1 + State_2.loc['CP/CV','State 2'] * (Mach_inlet**2) - f_dim) / (1 +␣
↪State_2.loc['CP/CV','State 2']\
                                                                            *␣
↪(Mach_outlet_1**2))

print("Pressure at Outlet of Heat Source =", round(P3/6894.8 , 3), "psia")
```

```
Pressure at Outlet of Heat Source = 2594.862 psia
```

[15]:
```
# Find the enthalpy at the outlet of the Heat source
Enth_3 = (Enth_2 * (1 + ((State_2.loc['CP/CV','State 2'] - 1) / 2) *␣
↪Mach_inlet**2 + q_dim)) / \
            (1 + ((State_2.loc['CP/CV','State 2'] - 1) / 2) *␣
↪(Mach_outlet_1**2))

Enth_3 # [J/kg]
```

[15]:
```
411010.8332202174
```

Outlet of Heat Source (State 3)

[16]:
```
# Using the new Pressure and enthalpy find the states of the fluid at the␣
↪outlet of Heat Source

State_3 = RP.REFPROPdll("CO2","PH","T;D;V;S;CP/CV;W;TCX;VIS;PRANDTL",␣
↪MASS_BASE_SI,0,0,P3,Enth_3,[1.0])

# Outputs will be placed into data frame for organization
State_3 = pd.DataFrame(State_3.Output[0:9],
            index = ['Temperature [K]', 'Density [kg/m^3]', 'Volume [m^3/kg]',␣
↪'Entropy [J/kg]',
                     'CP/CV', 'Speed of Sound', 'Thermal Cond. [W/(mK)]',␣
↪'Viscosity [Pa-s]', 'Prandtl'],
            columns = ['State 3'])

# Display the data frame
State_3
```

[16]:
```
                              State 3
Temperature [K]            361.571914
Density [kg/m^3]           481.379685
Volume [m^3/kg]              0.002077
Entropy [J/kg]            1605.371118
CP/CV                        2.861185
Speed of Sound             299.342200
Thermal Cond. [W/(mK)]       0.055138
Viscosity [Pa-s]             0.000036
Prandtl                      1.722122
```

[17]:
```python
# Add enthalpy to the data frame
State_2.loc['Enthalpy [J/kg]', 'State 2'] = Enth_2
State_3.loc['Enthalpy [J/kg]', 'State 3'] = Enth_3
```

[18]:
```python
# Find Velocity, Reynolds Number, and darcy friction factor (assuming smooth␣
↪pipe)
Velocity = 4 * m_dot * State_3.loc['Volume [m^3/kg]','State 3'] / (math.pi *␣
↪Tube_ID**2)
Reynolds = State_3.loc['Density [kg/m^3]','State 3'] * Velocity * Tube_ID /␣
↪State_3.loc['Viscosity [Pa-s]','State 3']
Darcy_f = (0.79 * math.log(Reynolds) - 1.64)**(-2)
```

[19]:
```python
# Using Estimated Length of Tubing connecting Heat Source and Engine
# find the amount of pressure drop caused by fanno flow
Length = 2.286 # [meters]

# Force acted on the wall of tube
Force = math.pi * Tube_ID * Darcy_f * State_3.loc['Density [kg/m^3]','State 3']␣
↪* (Velocity**2) * Length / 8

# Dimensionless Friction factor
f_dim = 4 * Force / (P3 * math.pi * Tube_ID**2)

# Heat Added into system (kW)
Q = 0

# Dimensionless heating factor
q_dim = Q * 1000 / (m_dot * State_3.loc['Enthalpy [J/kg]','State 3'])

# Inlet Mach Number of length of tubing
Mach_inlet = Velocity / State_3.loc['Speed of Sound','State 3']

# Formulation used to calculate Pressure drop (found in Adv. Fluid Mechanics␣
↪Textbook)
A_eq = ((Mach_inlet**2) * (1 + ((State_3.loc['CP/CV','State 3'] - 1) / 2) *␣
↪(Mach_inlet**2) + q_dim)) \
```

```
        / ((1 + State_3.loc['CP/CV','State 3'] * (Mach_inlet**2) - f_dim)**2)

# Find the positive outcome to the biquadratic Mach number
Mach_outlet_1 = math.sqrt((-1 * (1 - 2 * A_eq * State_3.loc['CP/CV','State 3'])\
                          + ((1 - 2 * A_eq * (State_3.loc['CP/CV','State 3'] +␣
 ↪1))**0.5))\
                         / ((State_3.loc['CP/CV','State 3'] - 1) - 2 * A_eq *␣
 ↪State_3.loc['CP/CV','State 3']**2))

# Find Outlet pressure caused by fanno flow (frictional loss)
P4 = P3 * (1 + State_3.loc['CP/CV','State 3'] * (Mach_inlet**2) - f_dim) / (1 +␣
 ↪State_3.loc['CP/CV','State 3']\
                                                                               *␣
 ↪(Mach_outlet_1**2))

print("Pressure at Inlet of Engine =", round(P4/6894.8 , 3), "psia")
```

```
Pressure at Inlet of Engine = 2591.134 psia
```

[20]:
```
# Find the enthalpy at the inlet of the Engine
Enth_4 = (State_3.loc['Enthalpy [J/kg]','State 3'] * (1 + ((State_3.loc['CP/
 ↪CV','State 3'] - 1) / 2) * Mach_inlet**2 + q_dim)) / \
            (1 + ((State_3.loc['CP/CV','State 3'] - 1) / 2) *␣
 ↪(Mach_outlet_1**2))

Enth_4 # [J/kg]
```

[20]:
```
411010.39266619587
```

Check order of states before continuing

[21]:
```
# Combine the data frames into one data frame for ease of use
sCO2_States = pd.concat([State_1, State_2, State_3], axis =1)

# Display the data frame to ensure proper layout
sCO2_States
```

[21]:

	State 1	State 2	State 3
Density [kg/m^3]	685.702611	685.353389	481.379685
Volume [m^3/kg]	0.001458	0.001459	0.002077
Enthalpy [J/kg]	331011.948444	331011.803977	411010.833220
Entropy [J/kg]	1374.687634	1374.825537	1605.371118
CP/CV	2.953841	2.957623	2.861185
Speed of Sound	373.917781	373.487733	299.342200
Thermal Cond. [W/(mK)]	0.074005	0.073968	0.055138
Viscosity [Pa-s]	0.000055	0.000055	0.000036
Prandtl	2.058781	2.061120	1.722122
Temperature [K]	NaN	333.106709	361.571914

[22]:
```
# Fill in the Missing data
sCO2_States.loc['Temperature [K]', 'State 1'] = T1
sCO2_States.loc['Pressure [Pa]', 'State 1'] = P1
sCO2_States.loc['Pressure [Pa]', 'State 2'] = P2
sCO2_States.loc['Pressure [Pa]', 'State 3'] = P3

# Display Data Frame
sCO2_States
```

[22]:

	State 1	State 2	State 3
Density [kg/m^3]	6.857026e+02	6.853534e+02	4.813797e+02
Volume [m^3/kg]	1.458358e-03	1.459101e-03	2.077362e-03
Enthalpy [J/kg]	3.310119e+05	3.310118e+05	4.110108e+05
Entropy [J/kg]	1.374688e+03	1.374826e+03	1.605371e+03
CP/CV	2.953841e+00	2.957623e+00	2.861185e+00
Speed of Sound	3.739178e+02	3.734877e+02	2.993422e+02
Thermal Cond. [W/(mK)]	7.400547e-02	7.396752e-02	5.513816e-02
Viscosity [Pa-s]	5.533361e-05	5.528707e-05	3.622383e-05
Prandtl	2.058781e+00	2.061120e+00	1.722122e+00
Temperature [K]	3.331500e+02	3.331067e+02	3.615719e+02
Pressure [Pa]	1.792648e+07	1.789489e+07	1.789105e+07

[23]:
```
# Reorder the Data Frame
sCO2_States = sCO2_States.reindex(["Pressure [Pa]", "Temperature [K]", 'Density␣
↪[kg/m^3]', 'Volume [m^3/kg]', 'Enthalpy [J/kg]',
                    'Entropy [J/kg]', 'CP/CV', 'Speed of Sound', 'Thermal Cond.
↪ [W/(mK)]', 'Viscosity [Pa-s]',
                    'Prandtl' ])
```

Inlet of Engine (State 4)

[24]:
```
# Using the new Pressure and enthalpy find the states of the fluid at the Inlet␣
↪of Engine

State_4 = RP.REFPROPdll("CO2","PH","T;D;V;S;CP/CV;W;TCX;VIS;PRANDTL",␣
↪MASS_BASE_SI,0,0,P4,Enth_4,[1.0])

# Outputs will be placed into data frame for organization
State_4 = pd.DataFrame(State_4.Output[0:9],
            index = ['Temperature [K]', 'Density [kg/m^3]', 'Volume [m^3/kg]',␣
↪'Entropy [J/kg]',
                    'CP/CV', 'Speed of Sound', 'Thermal Cond. [W/(mK)]',␣
↪'Viscosity [Pa-s]', 'Prandtl'],
            columns = ['State 4'])

# Display the data frame
State_4
```

[24]:
```
                              State 4
Temperature [K]            361.501662
Density [kg/m^3]           480.973226
Volume [m^3/kg]              0.002079
Entropy [J/kg]            1605.517639
CP/CV                        2.863499
Speed of Sound             299.087675
Thermal Cond. [W/(mK)]       0.055106
Viscosity [Pa-s]             0.000036
Prandtl                      1.723036
```

[25]:
```python
# Engine Parameters
mass_cylinder = State_4.loc['Density [kg/m^3]', 'State 4'] * .000308276
State_5_den = mass_cylinder / .000454574

# With Isentropic expansion
State_5_entr = State_4.loc['Entropy [J/kg]', 'State 4']
```

Outlet of Engine (State 5)

[26]:
```python
# Using the new density and entropy find the states of the fluid at the Outlet␣
↪of Engine

State_5 = RP.REFPROPdll("CO2","DS","P;T;V;H;CP/CV;W;TCX;VIS;PRANDTL",␣
↪MASS_BASE_SI,0,0,State_5_den,State_5_entr,[1.0])

# Outputs will be placed into data frame for organization
State_5 = pd.DataFrame(State_5.Output[0:9],
            index = ['Pressure [Pa]', 'Temperature [K]', 'Volume [m^3/kg]',␣
↪'Enthalpy [J/kg]',
                     'CP/CV', 'Speed of Sound', 'Thermal Cond. [W/(mK)]',␣
↪'Viscosity [Pa-s]', 'Prandtl'],
            columns = ['State 5'])

# Display the data frame
State_5
```

[26]:
```
                              State 5
Pressure [Pa]            8.417377e+06
Temperature [K]          3.137397e+02
Volume [m^3/kg]          3.065801e-03
Enthalpy [J/kg]          3.878968e+05
CP/CV                    6.565948e+00
Speed of Sound           2.017467e+02
Thermal Cond. [W/(mK)]   5.228222e-02
Viscosity [Pa-s]         2.415056e-05
Prandtl                  3.291732e+00
```

[27]:
```
# Combine the data frames into one data frame for ease of use
sCO2_States = pd.concat([sCO2_States, State_4, State_5], axis =1)

# Fill in the Missing data
sCO2_States.loc['Pressure [Pa]', 'State 4'] = P4
sCO2_States.loc['Enthalpy [J/kg]', 'State 4'] = Enth_4
sCO2_States.loc['Density [kg/m^3]', 'State 5'] = State_5_den
sCO2_States.loc['Entropy [J/kg]', 'State 5'] = State_5_entr

# Display the data frame
sCO2_States
```

[27]:
```
                         State 1       State 2       State 3  \
Pressure [Pa]           1.792648e+07  1.789489e+07  1.789105e+07
Temperature [K]         3.331500e+02  3.331067e+02  3.615719e+02
Density [kg/m^3]        6.857026e+02  6.853534e+02  4.813797e+02
Volume [m^3/kg]         1.458358e-03  1.459101e-03  2.077362e-03
Enthalpy [J/kg]         3.310119e+05  3.310118e+05  4.110108e+05
Entropy [J/kg]          1.374688e+03  1.374826e+03  1.605371e+03
CP/CV                   2.953841e+00  2.957623e+00  2.861185e+00
Speed of Sound          3.739178e+02  3.734877e+02  2.993422e+02
Thermal Cond. [W/(mK)]  7.400547e-02  7.396752e-02  5.513816e-02
Viscosity [Pa-s]        5.533361e-05  5.528707e-05  3.622383e-05
Prandtl                 2.058781e+00  2.061120e+00  1.722122e+00

                         State 4       State 5
Pressure [Pa]           1.786535e+07  8.417377e+06
Temperature [K]         3.615017e+02  3.137397e+02
Density [kg/m^3]        4.809732e+02  3.261790e+02
Volume [m^3/kg]         2.079118e-03  3.065801e-03
Enthalpy [J/kg]         4.110104e+05  3.878968e+05
Entropy [J/kg]          1.605518e+03  1.605518e+03
CP/CV                   2.863499e+00  6.565948e+00
Speed of Sound          2.990877e+02  2.017467e+02
Thermal Cond. [W/(mK)]  5.510589e-02  5.228222e-02
Viscosity [Pa-s]        3.618962e-05  2.415056e-05
Prandtl                 1.723036e+00  3.291732e+00
```

[28]:
```
# Find Velocity, Reynolds Number, and darcy friction factor (assuming smooth␣
↪pipe)
Velocity = 4 * m_dot * sCO2_States.loc['Volume [m^3/kg]','State 5'] / (math.pi␣
↪* Tube_ID**2)
Reynolds = sCO2_States.loc['Density [kg/m^3]','State 5'] * Velocity * Tube_ID /␣
↪sCO2_States.loc['Viscosity [Pa-s]','State 5']
Darcy_f = (0.79 * math.log(Reynolds) - 1.64)**(-2)
```

[29]:
```python
# Using Estimated Length of Tubing connecting Engine and Heat Exchanger
# find the amount of pressure drop caused by fanno flow
Length = 6.82 # [meters]

# Force acted on the wall of tube
Force = math.pi * Tube_ID * Darcy_f * sCO2_States.loc['Density [kg/m^3]','State␣
↪5'] * (Velocity**2) * Length / 8

# Dimensionless Friction factor
f_dim = 4 * Force / (sCO2_States.loc['Pressure [Pa]','State 5'] * math.pi *␣
↪Tube_ID**2)

# Heat Added into system (kW)
Q = 0

# Dimensionless heating factor
q_dim = Q * 1000 / (m_dot * sCO2_States.loc['Enthalpy [J/kg]','State 5'])

# Inlet Mach Number of length of tubing
Mach_inlet = Velocity / sCO2_States.loc['Speed of Sound','State 5']

# Formulation used to calculate Pressure drop (found in Adv. Fluid Mechanics␣
↪Textbook)
A_eq = ((Mach_inlet**2) * (1 + ((sCO2_States.loc['CP/CV','State 5'] - 1) / 2) *␣
↪(Mach_inlet**2) + q_dim)) \
        / ((1 + sCO2_States.loc['CP/CV','State 5'] * (Mach_inlet**2) -␣
↪f_dim)**2)

# Find the positive outcome to the biquadratic Mach number
Mach_outlet_1 = math.sqrt((-1 * (1 - 2 * A_eq * sCO2_States.loc['CP/CV','State␣
↪5'])\
                          + ((1 - 2 * A_eq * (sCO2_States.loc['CP/CV','State␣
↪5'] + 1))**0.5))\
                         / ((sCO2_States.loc['CP/CV','State 5'] - 1) - 2 * A_eq␣
↪* sCO2_States.loc['CP/CV','State 5']**2))

# Find Outlet pressure caused by fanno flow (frictional loss)
P6 = sCO2_States.loc['Pressure [Pa]','State 5'] * (1 + sCO2_States.loc['CP/
↪CV','State 5'] * (Mach_inlet**2) - f_dim) / (1 + sCO2_States.loc['CP/
↪CV','State 5']\
                                                                            *␣
↪(Mach_outlet_1**2))

print("Pressure at Inlet of Heat Exchanger =", round(P6/6894.8 , 3), "psia")
```

```
Pressure at Inlet of Heat Exchanger = 1205.343 psia
```

[30]:
```python
# Find the enthalpy at the inlet of the Heat Exchanger
Enth_6 = (sCO2_States.loc['Enthalpy [J/kg]','State 5'] * (1 + ((sCO2_States.
↪loc['CP/CV','State 5'] - 1) / 2) * Mach_inlet**2 + q_dim)) / \
            (1 + ((sCO2_States.loc['CP/CV','State 5'] - 1) / 2) *␣
↪(Mach_outlet_1**2))

Enth_6 # [J/kg]
```

[30]: 387843.81241681956

Inlet of Heat Exchanger (State 6)

[31]:
```python
# Using the new Pressure and enthalpy find the states of the fluid at the Inlet␣
↪of Heat Exchanger

State_6 = RP.REFPROPdll("CO2","PH","T;D;V;S;CP/CV;W;TCX;VIS;PRANDTL",␣
↪MASS_BASE_SI,0,0,P6,Enth_6,[1.0])

# Outputs will be placed into data frame for organization
State_6 = pd.DataFrame(State_6.Output[0:9],
            index = ['Temperature [K]', 'Density [kg/m^3]', 'Volume [m^3/kg]',␣
↪'Entropy [J/kg]',
                     'CP/CV', 'Speed of Sound', 'Thermal Cond. [W/(mK)]',␣
↪'Viscosity [Pa-s]', 'Prandtl'],
            columns = ['State 6'])

# Display the data frame
State_6
```

[31]:

	State 6
Temperature [K]	312.969473
Density [kg/m^3]	322.837925
Volume [m^3/kg]	0.003098
Entropy [J/kg]	1606.398556
CP/CV	6.677678
Speed of Sound	200.585930
Thermal Cond. [W/(mK)]	0.052346
Viscosity [Pa-s]	0.000024
Prandtl	3.336801

[32]:
```python
# Add enthalpy and Pressure to Data frame
State_6.loc['Enthalpy [J/kg]', 'State 6'] = Enth_6
State_6.loc['Pressure [Pa]', 'State 6'] = P6

# Combine the data frames into one data frame for ease of use
sCO2_States = pd.concat([sCO2_States, State_6], axis =1)

# Display the data frame
```

```
sCO2_States
```

[32]:

```
                           State 1       State 2       State 3  \
Pressure [Pa]          1.792648e+07  1.789489e+07  1.789105e+07
Temperature [K]        3.331500e+02  3.331067e+02  3.615719e+02
Density [kg/m^3]       6.857026e+02  6.853534e+02  4.813797e+02
Volume [m^3/kg]        1.458358e-03  1.459101e-03  2.077362e-03
Enthalpy [J/kg]        3.310119e+05  3.310118e+05  4.110108e+05
Entropy [J/kg]         1.374688e+03  1.374826e+03  1.605371e+03
CP/CV                  2.953841e+00  2.957623e+00  2.861185e+00
Speed of Sound         3.739178e+02  3.734877e+02  2.993422e+02
Thermal Cond. [W/(mK)] 7.400547e-02  7.396752e-02  5.513816e-02
Viscosity [Pa-s]       5.533361e-05  5.528707e-05  3.622383e-05
Prandtl                2.058781e+00  2.061120e+00  1.722122e+00

                           State 4       State 5       State 6
Pressure [Pa]          1.786535e+07  8.417377e+06  8.310597e+06
Temperature [K]        3.615017e+02  3.137397e+02  3.129695e+02
Density [kg/m^3]       4.809732e+02  3.261790e+02  3.228379e+02
Volume [m^3/kg]        2.079118e-03  3.065801e-03  3.097530e-03
Enthalpy [J/kg]        4.110104e+05  3.878968e+05  3.878438e+05
Entropy [J/kg]         1.605518e+03  1.605518e+03  1.606399e+03
CP/CV                  2.863499e+00  6.565948e+00  6.677678e+00
Speed of Sound         2.990877e+02  2.017467e+02  2.005859e+02
Thermal Cond. [W/(mK)] 5.510589e-02  5.228222e-02  5.234597e-02
Viscosity [Pa-s]       3.618962e-05  2.415056e-05  2.394502e-05
Prandtl                1.723036e+00  3.291732e+00  3.336801e+00
```

[33]:

```
# Find Velocity, Reynolds Number, and darcy friction factor (assuming smooth␣
↪pipe)
Velocity = 4 * m_dot * sCO2_States.loc['Volume [m^3/kg]','State 6'] / (math.pi␣
↪* Tube_ID**2)
Reynolds = sCO2_States.loc['Density [kg/m^3]','State 6'] * Velocity * Tube_ID /␣
↪sCO2_States.loc['Viscosity [Pa-s]','State 6']
Darcy_f = (0.79 * math.log(Reynolds) - 1.64)**(-2)
```

[34]:

```
# Using Estimated Length of Tubing used for Heat Exchanger
# find the amount of pressure drop caused by fanno flow & Heat loss
Length = 11.582 # [meters]

# Force acted on the wall of tube
Force = math.pi * Tube_ID * Darcy_f * sCO2_States.loc['Density [kg/m^3]','State␣
↪6'] * (Velocity**2) * Length / 8

# Dimensionless Friction factor
f_dim = 4 * Force / (sCO2_States.loc['Pressure [Pa]','State 6'] * math.pi *␣
↪Tube_ID**2)
```

```
# Heat into system (kW)
Q = -6.272

# Dimensionless heating factor
q_dim = Q * 1000 / (m_dot * sCO2_States.loc['Enthalpy [J/kg]','State 6'])

# Inlet Mach Number of length of tubing
Mach_inlet = Velocity / sCO2_States.loc['Speed of Sound','State 6']

# Formulation used to calculate Pressure drop (found in Adv. Fluid Mechanics␣
↪Textbook)
A_eq = ((Mach_inlet**2) * (1 + ((sCO2_States.loc['CP/CV','State 6'] - 1) / 2) *␣
↪(Mach_inlet**2) + q_dim)) \
        / ((1 + sCO2_States.loc['CP/CV','State 6'] * (Mach_inlet**2) -␣
↪f_dim)**2)

# Find the positive outcome to the biquadratic Mach number
Mach_outlet_1 = math.sqrt((-1 * (1 - 2 * A_eq * sCO2_States.loc['CP/CV','State␣
↪6'])\
                          + ((1 - 2 * A_eq * (sCO2_States.loc['CP/CV','State␣
↪6'] + 1))**0.5))\
                         / ((sCO2_States.loc['CP/CV','State 6'] - 1) - 2 * A_eq␣
↪* sCO2_States.loc['CP/CV','State 6']**2))

# Find Outlet pressure caused by fanno flow (frictional loss)
P7 = sCO2_States.loc['Pressure [Pa]','State 6'] * (1 + sCO2_States.loc['CP/
↪CV','State 6'] * (Mach_inlet**2) - f_dim) / (1 + sCO2_States.loc['CP/
↪CV','State 6']\
                                                                            *␣
↪(Mach_outlet_1**2))

print("Pressure at Outlet of Heat Exchanger =", round(P7/6894.8 , 3), "psia")
```

```
Pressure at Outlet of Heat Exchanger = 1180.109 psia
```

Outlet of Heat Exchanger (State 7)

[35]:
```
# Using the new Pressure and Temperature find the properties of the fluid at␣
↪the Outlet of Heat Exchanger
T7 = 36 + 273.15 # [K] will be controlled by cooling loop of Heat Exchanger

State_7 = RP.REFPROPdll("CO2","PT","D;V;H;S;CP/CV;W;TCX;VIS;PRANDTL",␣
↪MASS_BASE_SI,0,0,P7,T7,[1.0])

# Outputs will be placed into data frame for organization
State_7 = pd.DataFrame(State_7.Output[0:9],
```

```
            index = ['Density [kg/m^3]', 'Volume [m^3/kg]', 'Enthalpy [J/kg]',␣
↪'Entropy [J/kg]',
                     'CP/CV', 'Speed of Sound', 'Thermal Cond. [W/(mK)]',␣
↪'Viscosity [Pa-s]', 'Prandtl'],
            columns = ['State 7'])

# Display the data frame
State_7
```

[35]:

```
                          State 7
Density [kg/m^3]       410.242639
Volume [m^3/kg]          0.002438
Enthalpy [J/kg]     356323.446412
Entropy [J/kg]        1506.524735
CP/CV                   17.593695
Speed of Sound         185.233364
Thermal Cond. [W/(mK)]   0.078137
Viscosity [Pa-s]         0.000029
Prandtl                  8.213117
```

[36]:

```
# Add Pressure and Temperature to Data frame
State_7.loc['Pressure [Pa]', 'State 7'] = P7
State_7.loc['Temperature [K]', 'State 7'] = T7

# Combine the data frames into one data frame for ease of use
sCO2_States = pd.concat([sCO2_States, State_7], axis =1)

# Display the data frame
sCO2_States
```

[36]:

```
                            State 1       State 2       State 3  \
Pressure [Pa]          1.792648e+07  1.789489e+07  1.789105e+07
Temperature [K]        3.331500e+02  3.331067e+02  3.615719e+02
Density [kg/m^3]       6.857026e+02  6.853534e+02  4.813797e+02
Volume [m^3/kg]        1.458358e-03  1.459101e-03  2.077362e-03
Enthalpy [J/kg]        3.310119e+05  3.310118e+05  4.110108e+05
Entropy [J/kg]         1.374688e+03  1.374826e+03  1.605371e+03
CP/CV                  2.953841e+00  2.957623e+00  2.861185e+00
Speed of Sound         3.739178e+02  3.734877e+02  2.993422e+02
Thermal Cond. [W/(mK)] 7.400547e-02  7.396752e-02  5.513816e-02
Viscosity [Pa-s]       5.533361e-05  5.528707e-05  3.622383e-05
Prandtl                2.058781e+00  2.061120e+00  1.722122e+00

                            State 4       State 5       State 6       State 7
Pressure [Pa]          1.786535e+07  8.417377e+06  8.310597e+06  8.136615e+06
Temperature [K]        3.615017e+02  3.137397e+02  3.129695e+02  3.091500e+02
Density [kg/m^3]       4.809732e+02  3.261790e+02  3.228379e+02  4.102426e+02
```

```
Volume [m^3/kg]          2.079118e-03  3.065801e-03  3.097530e-03  2.437582e-03
Enthalpy [J/kg]          4.110104e+05  3.878968e+05  3.878438e+05  3.563234e+05
Entropy [J/kg]           1.605518e+03  1.605518e+03  1.606399e+03  1.506525e+03
CP/CV                    2.863499e+00  6.565948e+00  6.677678e+00  1.759369e+01
Speed of Sound           2.990877e+02  2.017467e+02  2.005859e+02  1.852334e+02
Thermal Cond. [W/(mK)]   5.510589e-02  5.228222e-02  5.234597e-02  7.813652e-02
Viscosity [Pa-s]         3.618962e-05  2.415056e-05  2.394502e-05  2.864207e-05
Prandtl                  1.723036e+00  3.291732e+00  3.336801e+00  8.213117e+00
```

[37]:
```
# Find Velocity, Reynolds Number, and darcy friction factor (assuming smooth␣
↪pipe)
Velocity = 4 * m_dot * sCO2_States.loc['Volume [m^3/kg]','State 7'] / (math.pi␣
↪* Tube_ID**2)
Reynolds = sCO2_States.loc['Density [kg/m^3]','State 7'] * Velocity * Tube_ID /␣
↪sCO2_States.loc['Viscosity [Pa-s]','State 7']
Darcy_f = (0.79 * math.log(Reynolds) - 1.64)**(-2)
```

[38]:
```
# Using Estimated Length of Tubing connecting Heat Exchanger and Compressor
# find the amount of pressure drop caused by fanno flow
Length = 2 # [meters]

# Force acted on the wall of tube
Force = math.pi * Tube_ID * Darcy_f * sCO2_States.loc['Density [kg/m^3]','State␣
↪7'] * (Velocity**2) * Length / 8

# Dimensionless Friction factor
f_dim = 4 * Force / (sCO2_States.loc['Pressure [Pa]','State 7'] * math.pi *␣
↪Tube_ID**2)

# Inlet Mach Number of length of tubing
Mach_inlet = Velocity / sCO2_States.loc['Speed of Sound','State 7']

# Formulation used to calculate Pressure drop (found in Adv. Fluid Mechanics␣
↪Textbook)
A_eq = ((Mach_inlet**2) * (1 + ((sCO2_States.loc['CP/CV','State 7'] - 1) / 2) *␣
↪(Mach_inlet**2))) \
        / ((1 + sCO2_States.loc['CP/CV','State 7'] * (Mach_inlet**2) -␣
↪f_dim)**2)

# Find the positive outcome to the biquadratic Mach number
Mach_outlet_1 = math.sqrt((-1 * (1 - 2 * A_eq * sCO2_States.loc['CP/CV','State␣
↪7'])\
                          + ((1 - 2 * A_eq * (sCO2_States.loc['CP/CV','State␣
↪7'] + 1))**0.5))\
                         / ((sCO2_States.loc['CP/CV','State 7'] - 1) - 2 * A_eq␣
↪* sCO2_States.loc['CP/CV','State 7']**2))
```

```python
# Find Outlet pressure caused by fanno flow (frictional loss)
P8 = (sCO2_States.loc['Pressure [Pa]','State 7']) * (1 + sCO2_States.loc['CP/
↪CV','State 7'] * (Mach_inlet**2) - f_dim) / (1 + sCO2_States.loc['CP/
↪CV','State 7'] * (Mach_outlet_1**2))

print("Pressure at Inlet of Compressor =", round(P8/6894.8 , 3), "psia")
```

```
Pressure at Inlet of Compressor = 1176.345 psia
```

[39]:
```python
# Find the enthalpy at the inlet of the Compressor
Enth_8 = (sCO2_States.loc['Enthalpy [J/kg]','State 7'] * (1 + ((sCO2_States.
↪loc['CP/CV','State 7'] - 1) / 2) * Mach_inlet**2)) / \
            (1 + ((sCO2_States.loc['CP/CV','State 7'] - 1) / 2) *␣
↪(Mach_outlet_1**2))

Enth_8 # [J/kg]
```

[39]:
```
356296.8110545878
```

Inlet of Compressor (State 8)

[40]:
```python
# Using the new Pressure and Enthalpy find the properties of the fluid at the␣
↪Inlet of Compressor

State_8 = RP.REFPROPdll("CO2","PH","T;D;V;S;CP/CV;W;TCX;VIS;PRANDTL",␣
↪MASS_BASE_SI,0,0,P8,Enth_8,[1.0])

# Outputs will be placed into data frame for organization
State_8 = pd.DataFrame(State_8.Output[0:9],
            index = ['Temperature [K]', 'Density [kg/m^3]', 'Volume [m^3/kg]',␣
↪'Entropy [J/kg]',
                     'CP/CV', 'Speed of Sound', 'Thermal Cond. [W/(mK)]',␣
↪'Viscosity [Pa-s]', 'Prandtl'],
            columns = ['State 8'])

# Display the data frame
State_8
```

[40]:

	State 8
Temperature [K]	308.987945
Density [kg/m^3]	409.358511
Volume [m^3/kg]	0.002443
Entropy [J/kg]	1506.643437
CP/CV	17.989866
Speed of Sound	184.720305
Thermal Cond. [W/(mK)]	0.078657
Viscosity [Pa-s]	0.000029

```
Prandtl                    8.357031
```

[41]:

```python
# Add Pressure and Enthalpy to Data frame
State_8.loc['Pressure [Pa]', 'State 8'] = P8
State_8.loc['Enthalpy [J/kg]', 'State 8'] = Enth_8

# Combine the data frames into one data frame for ease of use
sCO2_States = pd.concat([sCO2_States, State_8], axis =1)

# Display the data frame
sCO2_States
```

[41]:

```
                              State 1       State 2       State 3  \
Pressure [Pa]            1.792648e+07  1.789489e+07  1.789105e+07
Temperature [K]          3.331500e+02  3.331067e+02  3.615719e+02
Density [kg/m^3]         6.857026e+02  6.853534e+02  4.813797e+02
Volume [m^3/kg]          1.458358e-03  1.459101e-03  2.077362e-03
Enthalpy [J/kg]          3.310119e+05  3.310118e+05  4.110108e+05
Entropy [J/kg]           1.374688e+03  1.374826e+03  1.605371e+03
CP/CV                    2.953841e+00  2.957623e+00  2.861185e+00
Speed of Sound           3.739178e+02  3.734877e+02  2.993422e+02
Thermal Cond. [W/(mK)]   7.400547e-02  7.396752e-02  5.513816e-02
Viscosity [Pa-s]         5.533361e-05  5.528707e-05  3.622383e-05
Prandtl                  2.058781e+00  2.061120e+00  1.722122e+00

                              State 4       State 5       State 6  \
Pressure [Pa]            1.786535e+07  8.417377e+06  8.310597e+06
Temperature [K]          3.615017e+02  3.137397e+02  3.129695e+02
Density [kg/m^3]         4.809732e+02  3.261790e+02  3.228379e+02
Volume [m^3/kg]          2.079118e-03  3.065801e-03  3.097530e-03
Enthalpy [J/kg]          4.110104e+05  3.878968e+05  3.878438e+05
Entropy [J/kg]           1.605518e+03  1.605518e+03  1.606399e+03
CP/CV                    2.863499e+00  6.565948e+00  6.677678e+00
Speed of Sound           2.990877e+02  2.017467e+02  2.005859e+02
Thermal Cond. [W/(mK)]   5.510589e-02  5.228222e-02  5.234597e-02
Viscosity [Pa-s]         3.618962e-05  2.415056e-05  2.394502e-05
Prandtl                  1.723036e+00  3.291732e+00  3.336801e+00

                              State 7       State 8
Pressure [Pa]            8.136615e+06  8.110666e+06
Temperature [K]          3.091500e+02  3.089879e+02
Density [kg/m^3]         4.102426e+02  4.093585e+02
Volume [m^3/kg]          2.437582e-03  2.442846e-03
Enthalpy [J/kg]          3.563234e+05  3.562968e+05
Entropy [J/kg]           1.506525e+03  1.506643e+03
CP/CV                    1.759369e+01  1.798987e+01
Speed of Sound           1.852334e+02  1.847203e+02
```

```
Thermal Cond. [W/(mK)]  7.813652e-02  7.865692e-02
Viscosity [Pa-s]        2.864207e-05  2.857717e-05
Prandtl                 8.213117e+00  8.357031e+00
```